(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 July 2001 (26.07.2001)

PCT

(10) International Publication Number
WO 01/54369 A2

(51) International Patent Classification[7]: H04L 29/00

(21) International Application Number: PCT/CA01/00045

(22) International Filing Date: 24 January 2001 (24.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/490,778 24 January 2000 (24.01.2000) US

(71) Applicant: **NEWHEIGHTS SOFTWARE CORPORATION** [CA/CA]; 1006 Government Street, Victoria, British Columbia V8W 1X8 (CA).

(72) Inventors: **FISHER, Andrew**; 6-1221 Rockland Avenue, Victoria, British Columbia V8V 3J1 (CA). **LEVY, Michael, R.**; 2855 Lansdowne Road, Victoria, British Columbia V8R 3P8 (CA). **SWOVELAND, Jonathan**; 860 Maddison Street, Victoria, Britisch Columbia V8S 4C2 (CA). **MATTHEWS, Owen**; 2330 Esplanade, Victoria, British Columbia V8R 2W2 (CA).

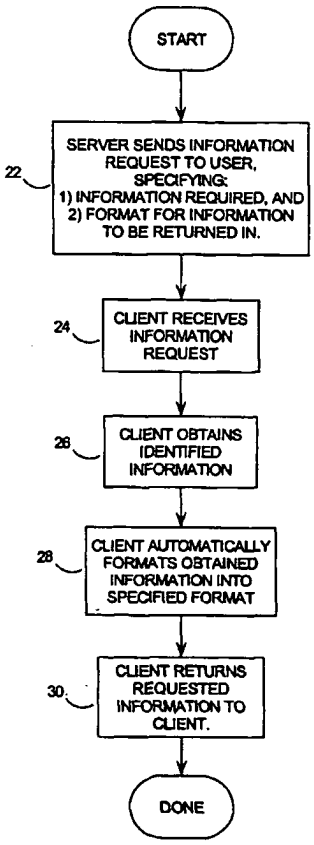
(74) Agents: **O'NEILL, Gary** et al.; Gowling Lafleur Henderson LLP, Suite 2600, 160 Elgin Street, Ottawa, Ontario K1P 1C3 (CA).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR COMPUTER NETWORK UPLOADING



(57) Abstract: The present invention relates generally to computer networks, and more specifically, to a system and method of rationalizing the different formats in which data is transferred between parties in a data communication network. For example, on the server side of a communication, information requests are sent which identify the information required and specify the format the information is to be returned in. A Media Agent on the client responds to the information request by obtaining and automatically formatting the identified information into the format requested by the server. If new drivers are required, the Media Agent may seek out those drivers from trusted sources on the Internet. The invention also allows other functionality, such as the server accessing physical devices on the client side, and the server interrogating the Media Agent for user preferences.

WO 01/54369 A2

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

System and Method for Computer Network Uploading

The present invention relates generally to computer networks, and more specifically, to a system and method of rationalizing the different formats in which data is transferred between parties in a data communication network.

Background of the Invention

It is well known that data communication networks such as the Internet, Wide Area Networks (WANs) and Local Area Networks (LANs), offer tremendously efficient means of organizing and distributing computerized data. These efficiencies have resulted in their widespread acceptance for both business and personal applications. For example, the Internet is now a common medium for operating online auctions, academic and public forums, distributing publications such as newspapers and magazines, and performing electronic commerce and electronic mail transactions. However, the existing methods and systems of implementing such services have limitations which hinder their efficiency and reliability, preventing even greater acceptance and use. Specifically, there is no simple and reliable way for parties in a data communication system to rationalize the format of a data transfer.

The Internet consists of a vast interconnection of computers, servers, routers, computer networks and public telecommunication networks which allows two parties to communicate via whatever entities happen to be interconnected at any particular time. Presently, the systems that make up the Internet comprise many different varieties of computer hardware and software. In general, this variety is not a great hindrance as the Internet is unified by a small number of standard transport protocols. These protocols transport data as simple packets, the nature of the packet contents being inconsequential to the transport itself. The difficulty lies in the receiving end which must determine how the contents of the received packet are to be read.

Technically, what distinguishes the Internet from other data networks is its use of a set of protocols called TCP/IP (Transmission Control Protocol/Internet Protocol). A protocol is a set of conventions or rules that govern transfer of data between hardware devices. The simplest protocols define only a hardware configuration while more complex protocols define timing, data formats, error detection and correction techniques and software structures. TCP/IP is the basic communication language or protocol of the Internet but can also be used as a communications protocol in the private networks called Intranet and in Extranets. TCP/IP uses the client/server

model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network.

The key advantage of a protocol like TCP/IP is that it allows a large network to function efficiently and that it offers a standardized means by which applications software can use that network. Of course, it does not define the contents or format of the contents of the data packets.

Higher layer application protocols use TCP/IP to access the Internet. Examples of higher protocols include the Hypertext Transfer Protocol (HTTP), the File Transfer Protocol (FTP), Telnet which allows logon to remote computers, and the Simple Mail Transfer Protocol (SMTP).

The Hypertext Transfer Protocol (HTTP) is the set of rules for exchanging files on the World Wide Web, including text, graphic images, sound, video, and other multimedia files. Relative to the TCP/IP suite of protocols, which are the basis for information exchange on the Internet, HTTP is an application protocol. HTTP also allows files to contain references to other files whose selection will elicit additional transfer requests (hypertext links).

Typically, the HTTP software on a Web server machine is designed to wait for HTTP requests and handle them when they arrive. A Web browser is an application program that runs on the client and provides a way to look at and interact with all the information on the World Wide Web. A Web browser uses HTTP to make requests of Web servers throughout the Internet on behalf of the Web browser user. Currently, most Web browsers are implemented as graphical user interfaces.

When the Web browser user enters file requests by either "opening" a Web file, typing in a Uniform Resource Locator (URL), or clicking on a hypertext link, the Web browser builds an HTTP request and sends it to the Internet Protocol address indicated by the URL. The HTTP software in the destination server machine receives the request and, after any necessary processing, the requested file is returned.

The client/server model has become one of the central ideas of network computing. In the typical client/server model **10** as presented in **Figure 1**, one server **12** is activated and awaits requests from multiple clients **14**. Typically, multiple client programs share the services of a common server program. Relative to the Internet, a Web browser is a client program that requests Web pages, files or other services from a Web server somewhere on the Internet.

Other program relationship models also exist including master/slave, where one program is in charge of all other programs, and peer-to-peer, where either of two programs is able to initiate a transaction. As well, multi-tiered models such as the three-tiered application server model as shown in **Figure 2**, are also common. In **Figure 2**, clients **14** use their Web browsers to request Web pages or files from the Web and/or application server **18**. The Web and/or application server **18** in turn, has access to a corporate database **20**. This arrangement provides a number of advantages over the simple client/server model of **Figure 1**, for example:

1. provides security by preventing direct access to all of the corporate database's **20** contents by clients **14**; and
2. allows processing to be performed on the data at the Web and/or application server **18**, rather than sending only raw data to the clients **14**.

This arrangement may also have various firewalls between parties and other security measures as known in the art. Although these different models may vary the roles of the applications software and higher level protocols, they typically use the same basic transfer protocols.

The application of *downloading* Web pages from a server to a client is generally straightforward, as the market is dominated by two major Web browsers with similar formats and functionality. However, there are a vast number of more complex applications which require *uploading* of data in numerous and complex formats. Examples of such complex applications include:

1. Filling out multi-page forms and forms that contain fields requiring the input of complex data such as images, data files and sound files or the input of multiples of such data. For example, Web sites often request personal data from clients such as their name, telephone number, address, employer and credit card numbers. This data may be used to set up a personal account, to send further information, or to allow access to certain services.
   The Internet would have far greater utility if more complex data could be easily and automatically uploaded such as: credit history for loan applications, employment history for job applications, and images for online auctions.
2. Security. If a higher level of security is required, Web sites will generally require a personal password or encryption key to be entered along with an account number. Generally, the more secure passwords and keys are too complex to be remembered by a user, so they must be stored and copied

electronically. Public encryption keys such as triple-DES, for example, require three 64-bit words.

Similarly, biometric data including images of the user's face, retina or fingerprint, or voiceprints of the user are also sometimes used as passwords in very high security applications, and must be generated in real time. Allowing the remote server to control the local device providing this biometric data would provide greater security, and reduced demands on resources because the client would not have to store the data.

3. Telephony over IP. There is currently a great deal of interest in using the Internet to transport voice data, avoiding the high cost of long distance telephone services. Typically, IP telephony is provisioned by the use of complementary encoding and decoding software at the client locations that are communicating. While telephones on the existing telephone system are almost universally intercompatible, the existing IP telephony solutions clearly are not. Unless both participants have the same software and version, which must be compatible with their respective platforms, they may not be able to communicate.

Traditionally, users format data to upload to a server in one of the following ways. Unfortunately, none of these methods is universal nor allows for reliable transfer of data:

1. manual key entry by user, into a form;
2. use of fixed and dedicated software;
3. manual downloading of drivers by the user;
4. attachment of plug-ins to a Web browser; or
5. embedded Web browser software.

Key entry, the use of fixed and dedicated software, and downloading of drivers are strictly manual methods of entry, while the use of plug-ins or embedded Web browser software offer small improvements in terms of automation. None of these approaches, however, offer a useful method of uploading data in view of the diverse software and hardware platforms that make up the Internet. Without such a method, the power of the Internet cannot be fully realized.

Most commonly, users enter data into forms presented by a Web page. If a user enters the incorrect data or enters the data with the incorrect syntax, this process fails. As well, there is a practical limitation to what data the user can input

manually. For example, the user cannot be expected to recall and enter a complex encryption code or public key accurately.

Alternatively, the user could upload generic files stored on his system using fixed and dedicated software. This approach may be used to perform quite sophisticated applications such as doing a retina scan in a fixed way, however this method is not flexible or reliable. For example, not all users would have enough understanding of their systems to do this, the file name entered may not always be accurate, and regardless, the data may not be in the correct software version or syntax. Even if all of these conditions are successfully met, the same exercise must be repeated each time the user uploads to a new server, or an old server which has upgraded its software.

Another common technique is for Web servers to place Hypertext links on their Web pages so that users may download the necessary drivers such as Adobe Acrobat or RealAudio. This still requires the user to have the skill to download the driver and configure it to his platform. As well, this requires that the proper version be downloaded; one that is compatible with the user's platform as well as that of the Web pages being sent.

Plug-in applications are programs that can be installed and used as part of a Web browser. A plug-in application is recognized automatically by the Web browser and its function is integrated into the main HTML (Hypertext Markup Language) file that is being presented. However, the Web browser must be reloaded in order to recognize a new plugin that has been downloaded. As well, using Web browser specific plug-ins also require a separate version of code for each Web browser and a re-start of the Web browser when the plug-in is modified or updated. HTML is the set of "markup" symbols or codes inserted in a file intended for display on a World Wide Web browser, which tells the Web browser how to display a Web page's words and images for the user. Other markup languages also exist such as XML and SGML.

Embedded Web browser software offers more advanced functionality than has traditionally been built into a Web browser. Examples of such software include XML, ActiveX, Java, JavaScript, HTML and Visual Basic Script.

These embedded solutions all share two major problems:

1. Require constant upgrading. Every time a new feature is added, the Web browser must be replaced with a new version which requires manually obtaining and installing the new software. This process is generally slow,

expensive, and frustrating, as installation and new compatibility problems may arise.

2. Not a universal solution. Being embedded in Web browsers prevents this from being a universal solution unless the same code is embedded in every browser. This is in direct conflict with the desire that users be allowed to have different platforms optimized to their particular applications.

Therefore, the use of embedded browser software only provides a partial solution to the problem. Several other specific limitations include, for example:

1. ActiveX has the ability to break out of the Web browser security model using signed code modules. ActiveX however, is limited to newer versions of Internet Explorer which immediately precludes the desired universality.

2. Java applets are small application programs that are compiled and run on the client and are commonly used to perform interactive animations, calculations, or other simple tasks. Like ActiveX, Java has the ability to break out of the Web browser security model using signed code modules.

However, Java does not provide an acceptable solution either. In order to load code outside of the Web browser, one must use a technology known as Java Native Interface (JNI). JNI is an API (applications programming interface) that determines how Java integrates with native code.

JNI allows a server to execute code on a user computer by utilizing signed code to break out of the Web browser's security model. Once this is done, system libraries can be loaded that contain the required functionality. The greatest limitation of JNI is that it is only supported by newer versions of Netscape, specifically Netscape with Java 1.1 support (Netscape 4.01+). As well, it requires a high degree of programming skill to interface directly with Java and other native languages.

3. CORBA™ (Common Object Request Broker Architecture) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an Object Request Broker (ORB). ORB support in a network of clients and servers on different computers means that a client program (which may itself be an object) can request services from a server program or object without having to

understand where the server is in a distributed network or what the interface to the server program looks like.

CORBA™ and DCOM provide the ability to talk to the server, but each object is specific to a program. That is, if a CORBA™ object is designed to capture a TWAIN image, that is what it will do and a separate program is required for each different option. For example, if a CORBA™ object is created to capture a TWAIN image, and upload it in bitmap format, a separate set of code is required if the image is to be captured in TWAIN and uploaded in JPG or GIF format.

4. COM (Component Object Model) is Microsoft™'s framework for developing and supporting program component objects. It is aimed at providing similar capabilities to those defined in CORBA™, a framework for the interoperation of distributed objects in a network that is supported by other major companies in the computer industry. Whereas Microsoft™'s OLE (Object Linking and Embedding) provides services for the compound document that users see on their display, COM provides the underlying services of interface negotiation, life cycle management (determining when an object can be removed from a system), licensing, and event services (putting one object into service as the result of an event that has happened to another object). COM can, for example, access local drivers.

OLE enables objects to be created with one application and then linked or embedded into a second application. Embedded objects retain their original format and links to the application that created them.

COM is generally supported by Windows™ and UNIX platforms, but a COM object created for a particular platform will only operate on that platform. COM is a binary standard that runs only on its intended platform, for example, if a COM object is designed for a UNIX platform, it will not run on any of the Windows™ platforms. Hence, in order to make the code universal, one would need to install a patch on most target machines or leave the user with no guarantee of delivery.

DCOM (Distributed Component Object Model) is an improvement over COM which can be instantiated remotely, but is still a binary standard and is platform specific. DCOM and COM, like CORBA™, require a different program for each version, of each application and platform.

5. Many types of server side services can be modelled in a satisfactory manner using XML, JavaScript, HTML, Dynamic HTML and Visual Basic Script, but none of these technologies addresses the ability to connect to client side devices and operating environments in a satisfactory manner.

Therefore, none of these methods provides a universal and efficient solution, let alone one that is platform independent. To summarize, several of the problems with these solutions are, on the Client side:

1. complex data may have to be entered manually;
2. software versions must be updated as required, with the risk of crashes if versions are incompatible;
3. different Web sites may require different software and configurations, so the user must have several Web browsers and must change Web browsers, drivers, plug-ins and preferences depending on the site being visited; and
4. old software must be discarded and new software purchased, which may be expensive, difficult to perform, slow, and frustrating.

Similar problems also arise on the Server side, for example:

1. processing cycles are consumed while compiling incoming data from various formats into the format the server desires;
2. can not upgrade software at the server without providing legacy support for clients who do not wish to upgrade, otherwise clients may be alienated;
3. concern about reliability and quality of service due to incompatibilities with client resources; and
4. code must be written in several different versions to support the prevailing embedded software products, allowing different users with their associated Web browsers, application programs and platforms. This requires multiple development efforts and also consumes additional resources

There is therefore a need for a method and system of rationalizing transferred data, which improves upon the problems described above. This design must be provided with consideration for ease of implementation and recognize the pervasiveness of existing infrastructure.

Summary of the Invention

It is therefore an object of the invention to provide method and system of uploading data, which improves upon the problems described above.

One aspect of the invention is broadly defined as a method of uploading information from a client to a server comprising the steps of: at the server: sending an information request to the client which identifies the information required and specifies the format the information is to be returned in; and at the client: receiving the information request; obtaining the identified information; automatically formatting the identified information into the specified format; and returning the formatted identified information to the server.

Another aspect of the invention is defined as a system for uploading information from a user to a server comprising: a server; a client; and a communication network interconnecting the server and the client; the server being operable to: send an information request to the client which includes a specification of the format the information is to be returned in; and the client being operable to: receive the information request; obtain the requested information; automatically format the requested information into the specified format; and return the formatted requested information to the server.

An additional aspect of the invention is defined as a computer data signal embodied in a carrier wave, the computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of: at the server: sending an information request to the client which identifies the information required and specifies the format the information is to be returned in; and at the client: receiving the information request; obtaining the identified information; automatically formatting the identified information into the specified format; and returning the formatted identified information to the server.

A further aspect of the invention is defined as a computer readable storage medium storing a set of machine executable code, the set of machine executable code being executable by a computer server to perform the steps of: at the server: sending an information request to the client which identifies the information required and specifies the format the information is to be returned in; and at the client: receiving the information request; obtaining the identified information; automatically formatting the identified information into the specified format; and returning the formatted identified information to the server.

Brief Description of the Drawings

These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings in which:

Figure 1 presents a block diagram of a client/server network as known in the prior art;

Figure 2 presents a block diagram of a three-tiered Web server and/or application server as known in the prior art;

Figure 3 presents a flow chart of an exemplary method in an embodiment of the invention;

Figure 4 presents a block diagram of an exemplary apparatus in an embodiment of the invention;

Figure 5 presents a bock diagram of a software architecture in an exemplary method of the invention;

Figures 6a and 6b present a flow chart of a preferred method of the invention; and

Figure 7 presents a block diagram of a software architecture in a preferred embodiment of the invention.

Description of the Invention

A method which addresses the objects outlined above, is presented as a flow chart in **Figure 3**. This flow chart presents a method of uploading information from a client to a server which is initiated at step **22** when the server sends an information request to the client which identifies the information required and also specifies the format that the information is to be returned in. The client receives the information request at step **24** and obtains the identified information at step **26** by accessing the necessary resources in the client computer. The client then automatically formats the identified information into the specified format at step **28** and returns the formatted identified information to the server at step **30**.

As outlined in the Background above, there are many instances in which it is desirable to upload information from the client to the server in a computer network. However, heretofore, there was no universal way for the client to rationalize the different formats in which the data could be uploaded.

The invention provides a method and system in which the server identifies the desired format of the upload, along with the request for the information. The client then receives this request and formats the data to comply with the request, prior to returning it to the server. The code in the information request which identifies the

desired format, may be parsed by the client by use of a header, identification code, or other manner as known in the art.

The client may obtain the requested information in a number of manners. In the simplest application, by querying the user for the data or location of the required files. Automatic methods of obtaining information will be outlined with respect to the preferred embodiment of the invention.

In this manner, the invention overcomes several of the problems described in the Background. For example, uploading of information was generally done in a manual manner in the past, either by means of the user entering data into a Web page based form, or by the user manually downloading the drivers it required to interact with the Web server. Attempts were made to automate the uploading of data, by use of plug-ins or Web browser-embedded software, but all of these methods suffered from the problems outlined in the Background.

With the invention, the server specifies the upload format, so the client is able to obtain and use the correct software and version, eliminating the incompatibility problems associated with different software or versions. This improves quality of service and reliability of the system. It also allows the Web servers to upgrade as new software becomes available, without having to worry that they have isolated themselves from their clients. Their client's software will automatically adapt to the upgrade, generally without the clients even being aware of the change.

The clients now may access any Web server, without concern for whether they have the correct software. Information requests from the Web server will specify exactly what software is required, and the client is free to either obtain the required software, or take a different line of action. In the preferred embodiment, a manner of automatically obtaining the new software will be described. If the client already has the required software, then its state will change transparently.

This implementation also allows the client to leverage their existing Web browser software. As described above, the use of embedded Web browser software would require the client to obtain new browser software with each change to any of the dozens of possible software formats commonly used. With the invention, the Web browser only serves to receive the information request for the client, and does not interpret it at all. Therefore, the client can continue to use an old version of his Web browser regardless of the requested upload format.

On the server side, the server no longer has to generate software code specific to each embedded Web browser language that a user may have. Only a single set of instructions are required for any particular functionality. This increases reliability due to the decreased complexity and decreases the server resources because only one set of software code is required. As well, this frees up server processing cycles because received packets do not have to be compiled from various incoming formats into a uniform format, because they are received in the format that is required.

An example of an apparatus upon which the invention may be performed is presented as a block diagram in **Figure 4**. This computer system **32** includes a display **34**, keyboard **36**, computer **38** and external devices **40**.

The computer **38** may contain one or more processors or microprocessors, such as a central processing unit (CPU) **42**. The CPU **42** performs arithmetic calculations and control functions to execute software stored in an internal memory **44**, preferably random access memory (RAM) and/or read only memory (ROM), and possibly additional memory **46**. The additional memory **46** may include, for example, mass memory storage, hard disk drives, floppy disk drives, magnetic tape drives, compact disk drives, program cartridges and cartridge interfaces such as that found in video game devices, removable memory chips such as EPROM, or PROM, or similar storage media as known in the art. This additional memory **46** may be physically internal to the computer **38**, or external as shown in **Figure 4**.

The computer system **32** also includes a communications interface **48** for communicating with an external network. Examples of the communications interface **48** can include a modem, a network interface such as an Ethernet card, or a serial or parallel communications port. Software and data transferred via communications interface **48** are in the form of signals which can be in an electronic, electromagnetic, optical or other form.

Input and output, to and from the computer **38**, is administered by the input/output (I/O) interface **50**. This I/O interface **50** administers control of the display **34**, keyboard **36**, external devices **40** and other such components of the computer system **32**.

The invention is described in these terms for convenience only. It would be clear to one skilled in the art that the invention may be applied to other computer or control systems **32**. Such systems would include all manner of appliances having

computer or processor control including telephones, cellular telephones, televisions, television set top units, lap top computers, personal digital assistants, industrial control equipment and automobiles.

Figure 5 presents a block diagram of an exemplary software architecture applied to an Internet environment, such as those presented in **Figures 1** or **2**. In this case, there is a Web server and/or application server **52**, as known in the art, which may have access to various databases or other internal or external resources. At the client location, a user employs a Web browser **54** to communicate bi-directionally with the Web server and/or application server **52** as known in the art. In addition to a Web browser **54** receiving Web pages over the Internet, the invention applies equally to an email program or application, or similar communication software.

The user also has a media agent **56** which detects information requests from the Web and/or application server **52** and is responsive to its contents in the manner of the invention described with respect to **Figure 3**. As noted above, the existing Web browser **54** is not responsive to such information request packets, and passes them on to the media agent **56** for interpretation. Media agent **56** may return responses via the Web browser **54**, or return them directly back to the Web and/or application server **52**. As well, the server **52** may direct requests to the media agent **56** itself, for example, to obtain information on the user's preferences. More details on these options are given in the discussion of the preferred embodiment which follows.

Preferred Embodiment

The preferred method of the invention is presented by means of a flow chart in **Figures 6a** and **6b**. As the preferred application of the invention is in a Web environment, the preferred method will be described in that context. It is understood that the invention may be equally applied to other data communication environments which would be known to one skilled in the art.

The method of the invention begins at step **58**, where the user requests a Web page or file from the server. As described above, this request is made using standard methods known in the art.

At step **60**, the server then returns a Web page to the user, with an embedded request for information in a high-level language and a dictionary of agents required.

The information request is embedded in the Web page by initiating the corresponding information request code with a character that will cause the user's Web browser to disregard it, rather than trying to read or display it and causing the Web browser to crash.

It is preferred that the embedded information request code commence with a comment code which will cause the Web browser to ignore it. As comment characters are relatively standard in the industry, this allows the method to be applied to any type or version of Web browser. Of course, dedicated or non-comment characters could be used, but this may compromise the universality of the implementation.

As well, it is not necessary for the embedded information request code to be written in a high level style, but it is preferred, as high level languages are easier to read. This makes the development, editing and troubleshooting of the embedded information request code easier.

It is also not necessary to include a dictionary of software agents required, as it is common to call upon agents, subroutines, function libraries and the like, as code is being compiled. However, when the invention is applied to Internet based applications, the time required to obtain software agents from a remote location may be greater than any of the other operations, so it is more efficient to source the software agents as early as possible so they are being downloaded while the balance of the code is compiling or running.

Of course, the balance of the Web page's contents has no relevance to the invention, so it may be coded with XML, ActiveX, Java, SGML, HTML or other similar languages that may become available from time to time.

When the user receives a Web page at step **62** using a standard Web browser **54**, the embedded information request code will also be received, but will not be displayed by the Web browser **54**. As noted above, the preferred manner of accomplishing this is to flag the information request code with characters, such as comment characters, which tell the Web browser **54** not to display it. The use of a standard Web browser **54** with standard Web browser controls minimizes the quantity of code needed to deploy the invention, as well as maximizing end user compatibility and leveraging the end user's existing software base.

After the user's Web browser **54** disregards the incoming embedded information request code, the information request code is passed to the user's filter at

step 64, which parses the code. The code executes as it is parsed in the same manner as a regular software program executes. The code itself is not passed to the memory cache or to the media engine, but may execute commands that transfers data or instructions to either of these components.

The media agent 56 determines at step 66, what software agents are needed and determines whether they are local, or have to be sourced remotely. This may be done by maintaining a table of local software agents and their locations, or by searching for them on the local storage media.

If new software agents are required, they are obtained at step 68. In the preferred embodiment of the invention, the media agent 56 will be pre-programmed with user preferences including where to look for software agents. It is expected that software agents will be stored in one or more central registries which provide standard benefits to the user such as security and help assistance.

It would be expected that the market would prefer to dynamically download new software agents from a reliable and predetermined source rather than facing the security and reliability risks of an open market, however, the invention may be applied in an open manner.

Now having all the necessary software agents, the media agent 56 obtains the data required at step 70. Preferably, this step is also automated, but addressing the location of certain data files may require user interaction. For example, one would expect clients to have standard and often requested data files such as those usually required to set up a Web account. This file would contain data such as name, address, email address, password data and biometric data.

If all the information that a particular Web server has requested is not contained in the automatically accessed files, then the user may be queried at step 72, for either the location of the required data, or the data itself.

Once all of the required data locations have been identified, those files are indexed at step 74 and the necessary information obtained. This information is then transferred to the memory cache at step 76 and formatted into the requested data structure.

A dynamic link library (DLL) may be used to execute the formatting agents on the data. A DLL is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer. For example, the small

program that allows a larger program to communicate with a specific device such as a printer or scanner is often packaged as a DLL file.

The use of DLL files provides the added advantage that they are not loaded into random access memory (RAM) together with the main program, saving space in the RAM. A DLL file is not loaded and run until it is required. DLL files are dynamically linked with the program that uses them during program execution rather than being compiled with the main program. UNIX and MacIntosh™ platforms do not use DLLs but have similar utilities that would be known to one skilled in the art.

Steps **70** through **76** are described with respect to the simple example of a data form, but may also be applied to far more complex applications. For example, the embedded information request code may contain TWAIN code which directly controls an image input device.

At step **78**, the media agent then returns the structured data to the server **52**. As will be described with respect to **Figure 7**, this return may be performed directly from the media agent **56** to the server **52**, or via the Web browser **54**.

The data is transferred from the media agent to the browser and back using HTTP protocol for several reasons. Firstly, because of HTTP's widespread use, a large number of software developers are familiar with it and how to develop code for it. Secondly, and more importantly, the use of HTTP allows the media agent **56** to simply look like another server to the Web browser **54**. This allows the media agent **56** to act outside the security model of the Web browser **54**, so it will be allowed to access devices on the client's system. Typically, Web browsers **54** are not allowed such root access to prevent malicious or defective code arriving via the communications network, from damaging the client's system.

This secure "sandbox" is expanded by use of the invention without fear of security breaches provided that secure software agents are used. The invention allows software agents to have root access, and if the software agents themselves are controlled and secure, the security of the system is not comprised. As will be described hereinafter, it is preferred to create a central repository of such controlled and secure software agents which is available to users over the Internet. Although users may obtain software agents from any supplier, it is expected that they will prefer to rely on such repositories to ensure that only secure software agents are used.

Again, fixed applications have been available in the past that have also had such root access over a communications network, but these solutions have been limited to fixed applications with predetermined formats. The invention provides for root access that is inherently generic and flexible.

In order to utilize the lowest common denominator user interface (HTML), the invention applies a resolution engine capable to taking as inputs the output from a HTTP post call, and a standard HTML template and producing a resultant form. An HTTP post call is an HTTP method used to send a block of data to a server to be processed in some way and is well known in the art. Furthermore, client side data can also be seeded into the template by adding it as though it came from the HTTP post input stream.

Standard Internet security methods such as firewalls, password verification, authentication, and encryption techniques may be applied to the invention. As one must ask software agents to enter their machine, simple authentication and trust may be used. Generally, a user is breaking out of the sandbox only once, and bypassing security checks once downloading.

Figure 7 presents a block diagram of a software architecture for implementing the invention in a preferred manner of the invention. The Web server and/or application server **52** and Web browser **54** may be the same as those described with respect to **Figure 5**. In the preferred embodiment, the media agent **56** comprises the architecture shown, which also has access to the devices, files and services **80** of the local computer system.

These devices, files and services **80** may include those resources described with respect to **Figure 4** above. This would include, for example, the display **34**, keyboard **36**, CPU **42**, internal memory **44**, additional memory **46**, communications interface **48** and input/output (I/O) interface **50**. Allowing the Web server and/or application server **52** access to such devices allows far more elaborate applications to be implemented, as well as saving client resources, work and time. For example:

1. it allows a Web server to have direct control over a user's digital camera so that it can authenticate the user visually. Using a traditional system, the Web server would not be able to prevent an unauthorized user from uploading a previously stored image file that belonged to the authorized user.

2. it allows a Web server to request a complex encryption key or digital signature which is stored on a hard drive, automatically seeking out the file without the user's interaction.

The storage location for new software agents **82** has been shown as a separate block from the perspective of the software, but it is understood that software agents may be stored on the same internal memory **44** or additional memory **46** on which the other local software entities are stored. Of course, new software agents are preferably stored at a central depository accessible via the Internet, as described above. This central repository may even reside on the same server **52** that the user is accessing.

The media agent **56** itself includes a filter **84** which removes the information requests from the received Web pages and pass the information format and contents to the media engine **88**. The media engine **88** itself includes an agent update and loader **90**, and a number of software agents **92**. These software agents **92** may include, for example:

1. CORBA™, COM™ or other embedded software languages. In the Background, it was explained that the current embedded software languages are not flexible or practical as a method of coordinating the format of data being transmitted. However, the invention allows data to be wrapped into a CORBA™ or COM™ object for uploading.
2. TWAIN is a widely-used standard for registering and launching drivers that are used to capture images as a bitmap from digital imaging devices. For example, TWAIN is commonly included with scanner software packages, running between the scanner hardware and an application, to convert scanner data directly into an application (such as PhotoShop™) where the image is to be worked upon. Without TWAIN, one would have to close an application that was open, open a special application to receive the image, and then move the image to the application where it was to be worked with.
3. Capturing voice streams using a microphone and converting to sound formats such as MP3™, Wave™, RealAudio™ and MPEG.
4. Word processing formats such as Microsoft™ Word™ and Corel™ WordPerfect™.
5. Output formats for printers.
6. Video drivers.

As the data are collected, they are stored in the memory cache **88**. The stored data are then passed to a format block **94** which executes the format agents on the cached data, to generate the data file requested by the Web server **52**. As noted above, the structured data may be returned either directly to the Web server **52**, or via the micro Web server **96**, through the browser **54**. This is possible because the data packets are returned using HTTP, so the Web browser **54** assumes that those data packets have been sent from another server and simply passes them on.

As explained above, the information request code is embedded in the incoming Web pages so that it will pass through the Web browser. The Web browser is avoided because existing Web browser embedded technologies like XML, ActiveX™, Java, JavaScript, HTML and Visual Basic Script do not have the functionality to provide a flexible solution. As described in the background these embedded technologies provide only a partial solution to the problem, each of these methods having limitations that prevent its widespread use. However, the Web browser **54** does provide a very useful graphic user interface (GUI) for presentation of data to the user, and for control of information inside the browser.

Additional Communication Examples

The invention was described with respect to an example presented in **Figures 6A** and **6B**, however, it would be clear to one skilled in the art that the bidirectional intercommunication provided by the invention is flexible enough to be applied to many other situations. Several examples are described hereinafter with respect to the Server **52**, Application **54** and Media Agent **56** presented in **Figure 5**, but this is by no means a complete list:

1. Application **54** to Server **52**
   a. In an initial session, the Server **52** could respond to a request for a web page received from the Application **54** by verifying access and returning to the Application **54** a lowest common denominator agent (LCDA) for a given protocol.
   b. In subsequent sessions, or service update sessions, the Server **52** could respond to a request for a web page received from the Application **54** by verifying access, processing the request, and returning a formatted response to the Application **54**.

2. Application **54** to Media Agent **56**

a. Communications of this type allow the Application **54** to activate local devices and other resources under the control of the Media Agent **56**. Such device control is well described with respect to the preferred embodiment above.

b. As well, the Application **54** may access the Media Agent **56** to perform network services. For example, the Application **54** may consist of legacy wordprocessing software that otherwise would not have access to the Server **52**. However, with the invention, the wordprocessing Application **54** could direct information to the Media Agent **56** as a default printer, and the Media Agent **56** could convert the information from a postscript format to COM, XML or Corba or other format to the Server **52**.

3. Media Agent **56** to Application **54**

a. In response to a service request, the Media Agent **56** may negotiate with other Applications **54** to achieve the desired objective. The other Applications **54** may include third party applications, appliances, services and devices.

b. Service delivery from the Media Agent **56** to the Application **54** may require data transformation from the service's specific protocol to a protocol compatible with the Application **54**.

For example, the Server **52** could receive a request from the user, via his browser which is an Application **54**, to perform a search using the keyword "Voltaire". The Server **52** responds to the request from the user, and the user's Media Agent **56** can receive the response before passing it on to the application. Based on known user preferences, the Media Agent **56** can determine whether the user is likely looking for information on the philosopher and author named "Voltaire", or the rock band of the same name, and resubmit the request modified for the users preferences.

With the response from the Media Agent **56**, the Server **52** may then perform a search that is tailored to the user's preferences and history, and serve the new response, without interruption from the media agent, back to the application.

4. Media Agent **56** to Server **52**
   a. This communication allows the Media Agent **56** to request services directly from the Server **52**. For example, if the Media Agent **56** requires a new software agent **92** to comply with a service request it has received in some manner, it may contact the Server **52** directly to obtain the necessary software agent **92**.
   b. As well, service update sessions may be performed between the Media Agent **56** and the Server **52**.
5. Server **52** to Media Agent **56**
   a. The Server **52** may provide services directly to the Media Agent **56**, for example, by providing a new software agent **92** as described above.
   b. The Server **52** may request services directly from the Media Agent **56** itself. As the Media Agent **56** and the Application **54** share the same IP (Internet Protocol) address, the TCP/IP stack can be read by the Media Agent **56** when a Server **52** responds to the request from the Application **54**. The Media Agent **56** can intervene before the data packets are sent to the Application **54**, hence the Server **52** is contacting the user's Media Agent **56** directly, requesting information about the user's preferences, history or other useful data to complete a user's request or transaction.

Applications

The invention may be applied to standard Internet client/server applications such as filling out forms and authenticating users. However, an unlimited number of new applications become practical with the added functionality of the invention, for example:

1. television set top boxes for television over IP, with the functionality of the invention embedded into a chipset;
2. remote and automatic, troubleshooting and maintenance of software and hardware systems;
3. telephony over IP where a software codec agent performs the encoding and decoding of the voice packets;
4. more advanced and reliable security techniques, for example, controlling biometric input devices directly to prevent a security breech; and

5. remote medical monitoring and/or diagnosis, either between hospitals or between patients at home, and their doctors or hospitals.

Examples have been shown to demonstrate various aspects of the invention, but the number of variations is by no means complete. Comparable implementations could be made for any computer network device, including personal digital assistants, cellular telephones, fax machines, pagers, point of sale computers, local area networks or private branch exchanges. While particular embodiments of the present invention have been shown and described, it is clear that changes and modifications may be made to such embodiments without departing from the true scope and spirit of the invention.

The method steps of the invention may be embodied in sets of executable machine code stored in a variety of formats such as object code or source code. Such code is described generically herein as programming code, or a computer program for simplification. Clearly, the executable machine code may be integrated with the code of other programs, implemented as subroutines, by external program calls or by other techniques as known in the art.

The embodiments of the invention may be executed by a computer processor or similar device programmed in the manner of method steps, or may be executed by an electronic system which is provided with means for executing these steps. Similarly, an electronic memory means such computer diskettes, CD-Roms, Random Access Memory (RAM), Read Only Memory (ROM) or similar computer software storage media known in the art, may be programmed to execute such method steps. As well, electronic signals representing these method steps may also be transmitted via a communication network.

WHAT IS CLAIMED IS:

1. A method of uploading information from a client to a server comprising the steps of:

at said server:

sending an information request to said client which identifies the information required and specifies the format said information is to be returned in; and

at said client:

responding to receipt of said information request by:

obtaining said identified information;

automatically formatting said identified information into said specified format; and

returning said formatted identified information to said server.

2. The method of claim 1 further comprising the steps of:

at said client:

identifying a software agent which may execute to format said requested information into said specified format; and

responding to said software agent not being available locally, by downloading said software agent.

3. The method of claim 2 wherein:

said step of sending includes sending a dictionary of software agents required to said client; and

said step of identifying includes identifying said required software agents by reviewing said dictionary.

4. The method of claim 3, wherein the step of downloading includes downloading said software agent from a pre-determined central repository.

5. The method of claim 2, wherein said step of obtaining said requested information includes locating said requested information locally.

6. The method of claim 5, wherein said step of locating said requested information includes indexing a pre-programmed local database.

7. The method of claim 2, further comprising the step of:

at said client:

accessing local devices to comply with requirements of said information request.

8. The method of claim 7, wherein preferences regarding interaction with local devices are pre-programmed, enabling them to provide data for uploading.

9. The method of claim 2, wherein said step of receiving said information request includes receiving said information request using a standard Web browser.

10. The method of claim 9, wherein:

said step of sending includes sending said information request as software code embedded in a Web page in a manner that said standard Web browser will not attempt to read; and

said step of receiving further includes filtering said embedded information request from said Web page and passing said filtered instruction to an agent operable to execute said information request code.

11. The method of claim 10, wherein said agent further comprises a micro Web server for uploading data packets to said browser, for graphic user interface (GUI) control and information control inside the browser.

12. The method of claim 9, wherein said agent looks like another server to said standard Web browser, allowing said agent to avoid security restrictions in said standard Web browser.

13. The method of claim 2, wherein said agent may query said Web user to input data through a standard Web browser.

14. A system for uploading information from a client to a server comprising:

a server;

a client; and

a communication network interconnecting said server and said client;

said server being operable to:

send an information request to said client which includes a specification of the format said information is to be returned in; and

said client being operable to:

receive said information request;

obtain said requested information;

automatically format said requested information into said specified format; and

return said formatted requested information to said server.

15. A system as claimed in claim 15, wherein said server is an application server.

16. A computer data signal embodied in a carrier wave, said computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of claim 1.

17. A computer readable storage medium storing a set of machine executable code, said set of machine executable code being executable by a computer to perform the steps of claim 1.

18. A method of uploading information from a client to a server comprising the steps executed, at said server, of:

sending an information request to said client which identifies the information required and specifies the format said information is to be returned in, whereby said client may responding to receipt of said information request by:

obtaining said identified information;

automatically formatting said identified information into said specified format; and

returning said formatted identified information to said server.

19. The method of claim 18 wherein said step of sending includes sending a dictionary of software agents required.

20. The method of claim 19 wherein said step of sending includes sending instructions to access and control local devices.

21. The method of claim 20, wherein said step of sending includes sending said information request as software code embedded in a Web page in a manner that said standard Web browser will not attempt to read.

FIGURE 1

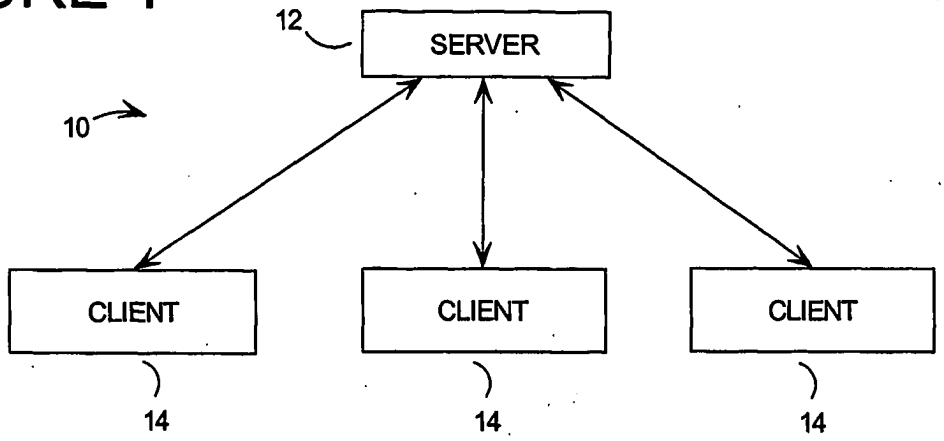
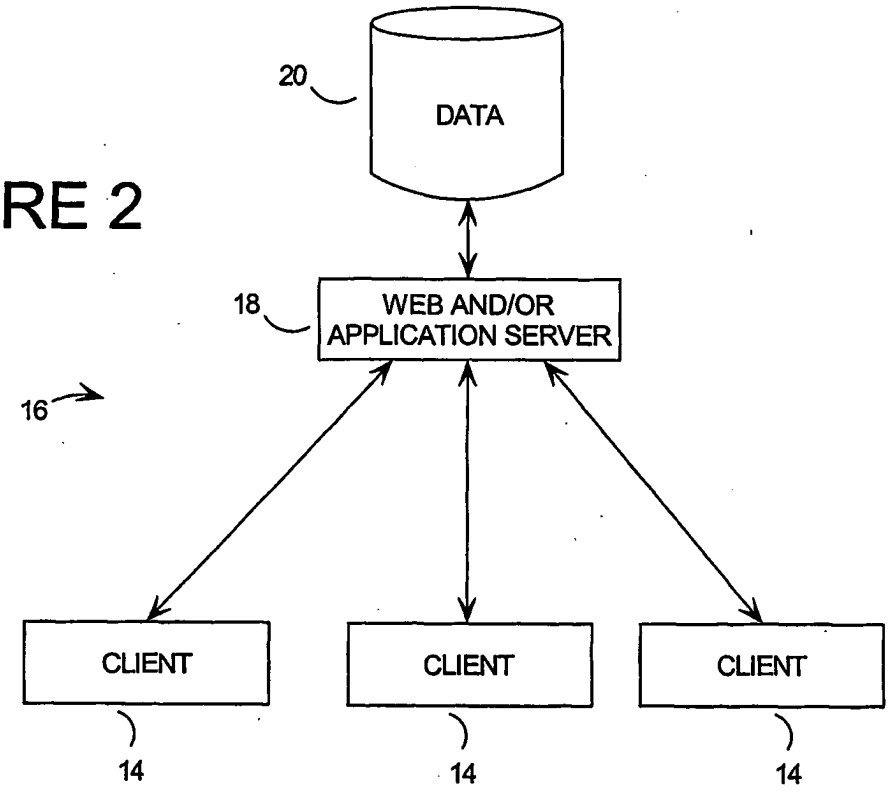


FIGURE 2



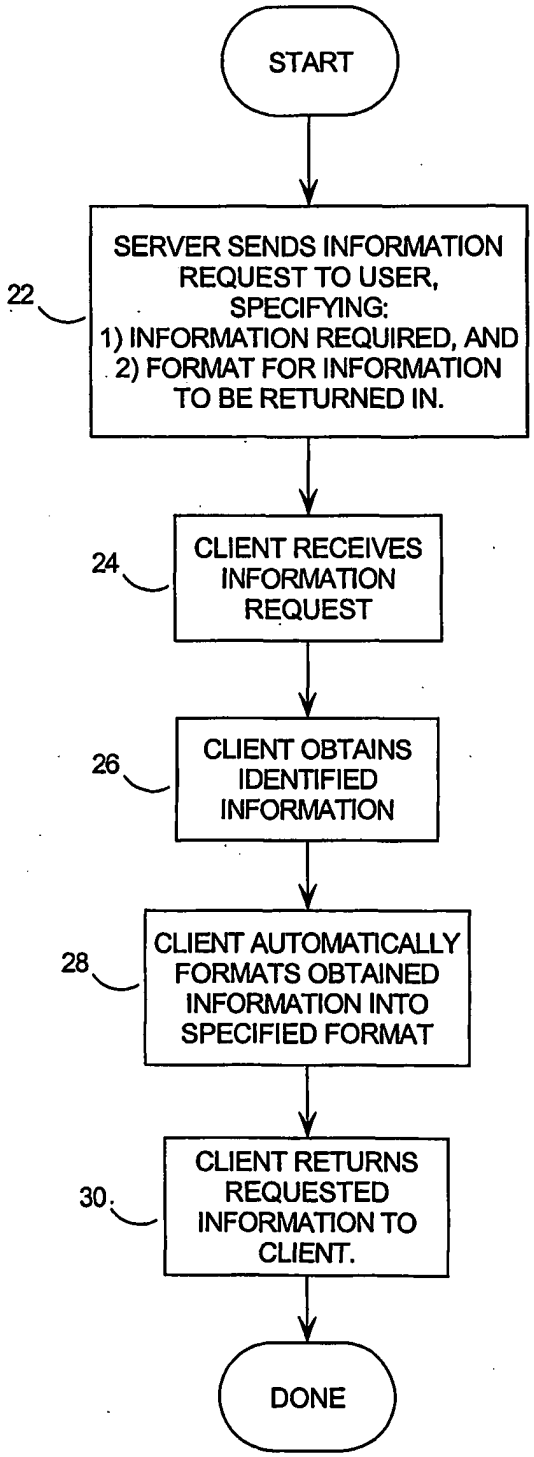
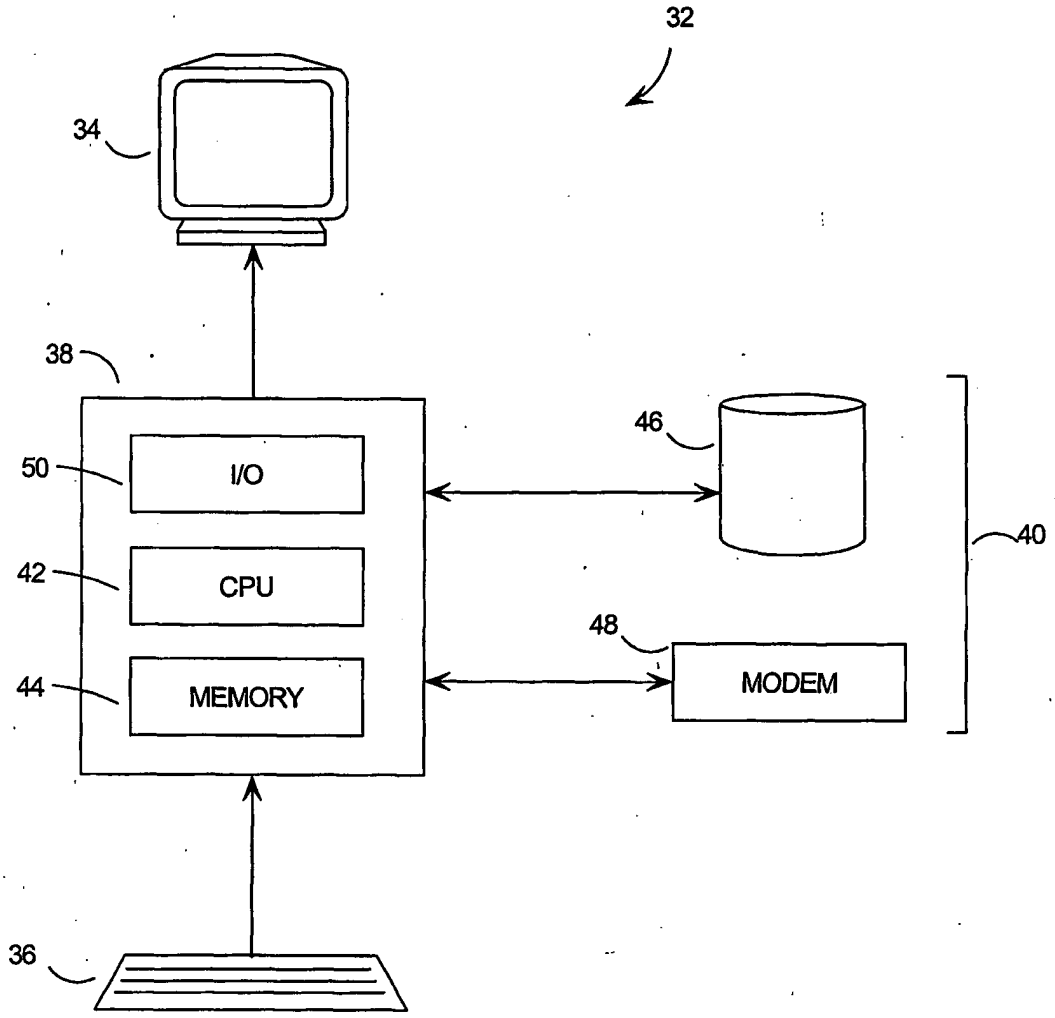

START
SERVER SENDS INFORMATION REQUEST TO USER, SPECIFYING: 1) INFORMATION REQUIRED, AND 2) FORMAT FOR INFORMATION TO BE RETURNED IN.
22
CLIENT RECEIVES INFORMATION REQUEST
24
CLIENT OBTAINS IDENTIFIED INFORMATION
26
CLIENT AUTOMATICALLY FORMATS OBTAINED INFORMATION INTO SPECIFIED FORMAT
28
CLIENT RETURNS REQUESTED INFORMATION TO CLIENT.
30
DONE

FIGURE 3

FIGURE 4



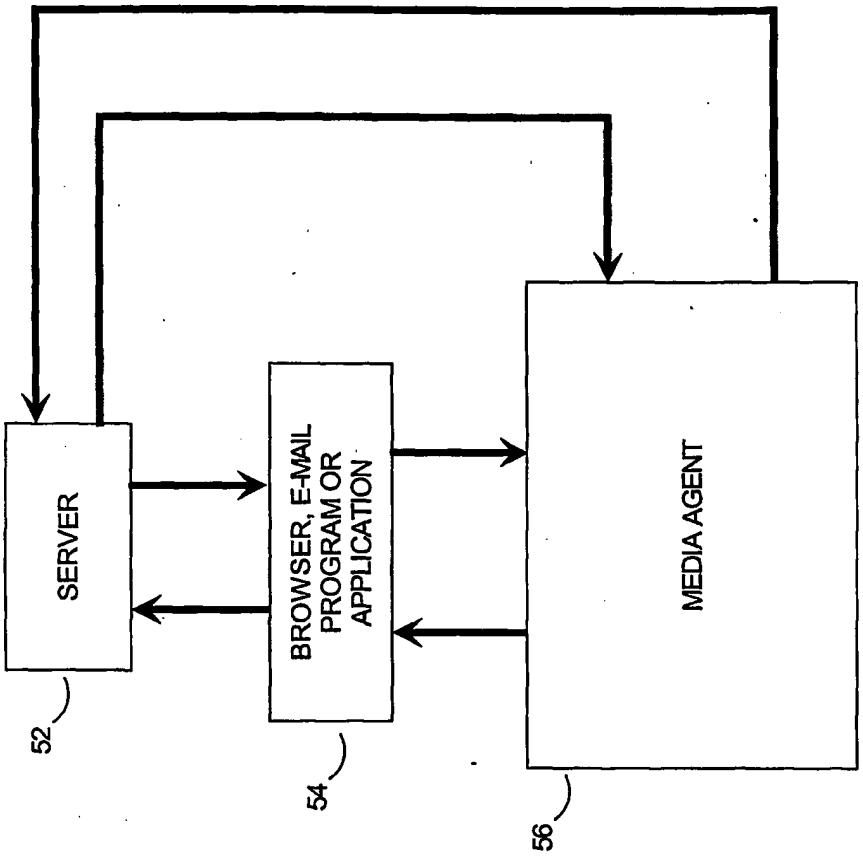

SERVER
BROWSER, E-MAIL PROGRAM OR APPLICATION
MEDIA AGENT
52
54
56

FIGURE 5

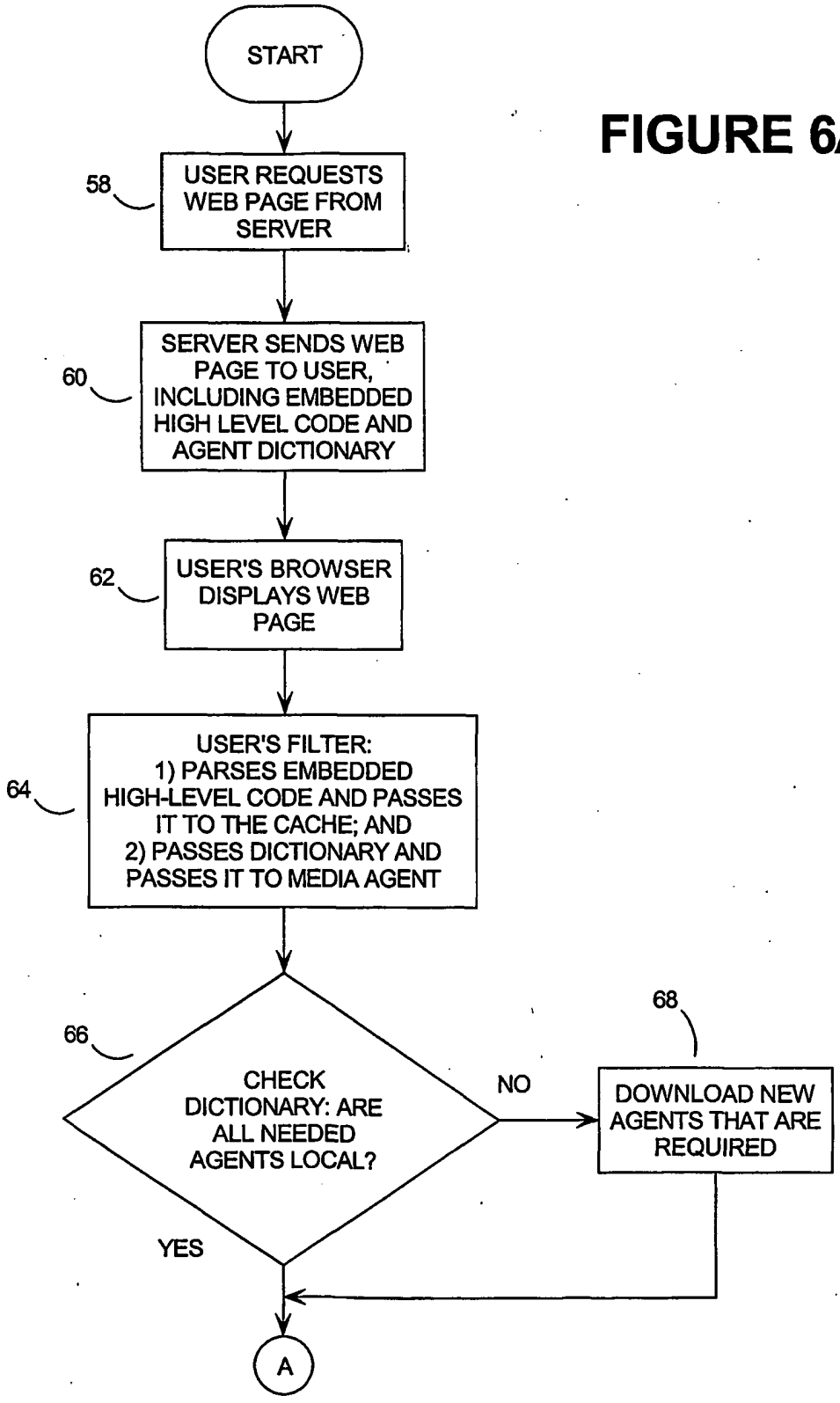
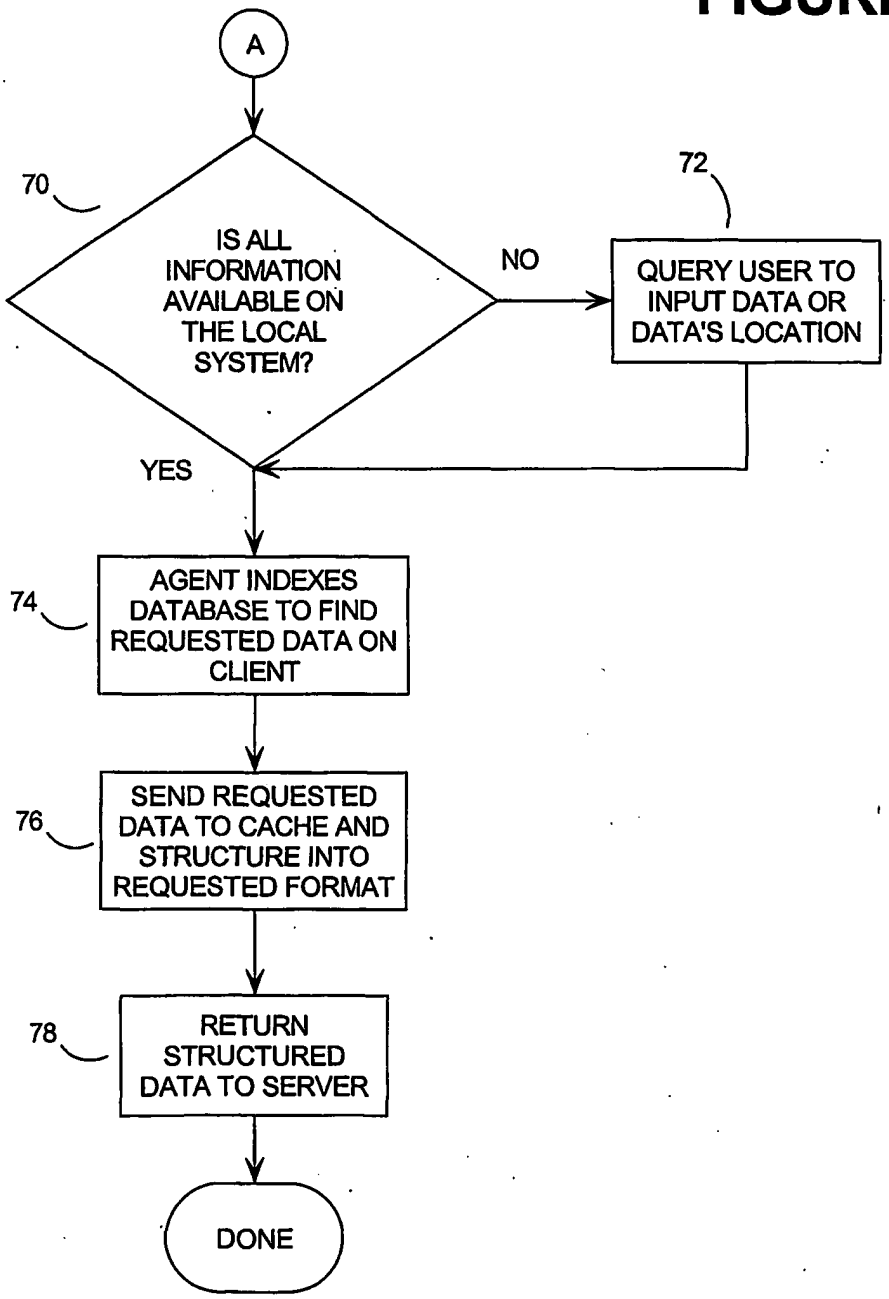

START
58
USER REQUESTS WEB PAGE FROM SERVER
60
SERVER SENDS WEB PAGE TO USER, INCLUDING EMBEDDED HIGH LEVEL CODE AND AGENT DICTIONARY
62
USER'S BROWSER DISPLAYS WEB PAGE
64
USER'S FILTER: 1) PARSES EMBEDDED HIGH-LEVEL CODE AND PASSES IT TO THE CACHE; AND 2) PASSES DICTIONARY AND PASSES IT TO MEDIA AGENT
66
CHECK DICTIONARY: ARE ALL NEEDED AGENTS LOCAL?
NO
68
DOWNLOAD NEW AGENTS THAT ARE REQUIRED
YES
A

FIGURE 6A

FIGURE 6B


A
70
IS ALL INFORMATION AVAILABLE ON THE LOCAL SYSTEM?
NO
72
QUERY USER TO INPUT DATA OR DATA'S LOCATION
YES
74
AGENT INDEXES DATABASE TO FIND REQUESTED DATA ON CLIENT
76
SEND REQUESTED DATA TO CACHE AND STRUCTURE INTO REQUESTED FORMAT
78
RETURN STRUCTURED DATA TO SERVER
DONE

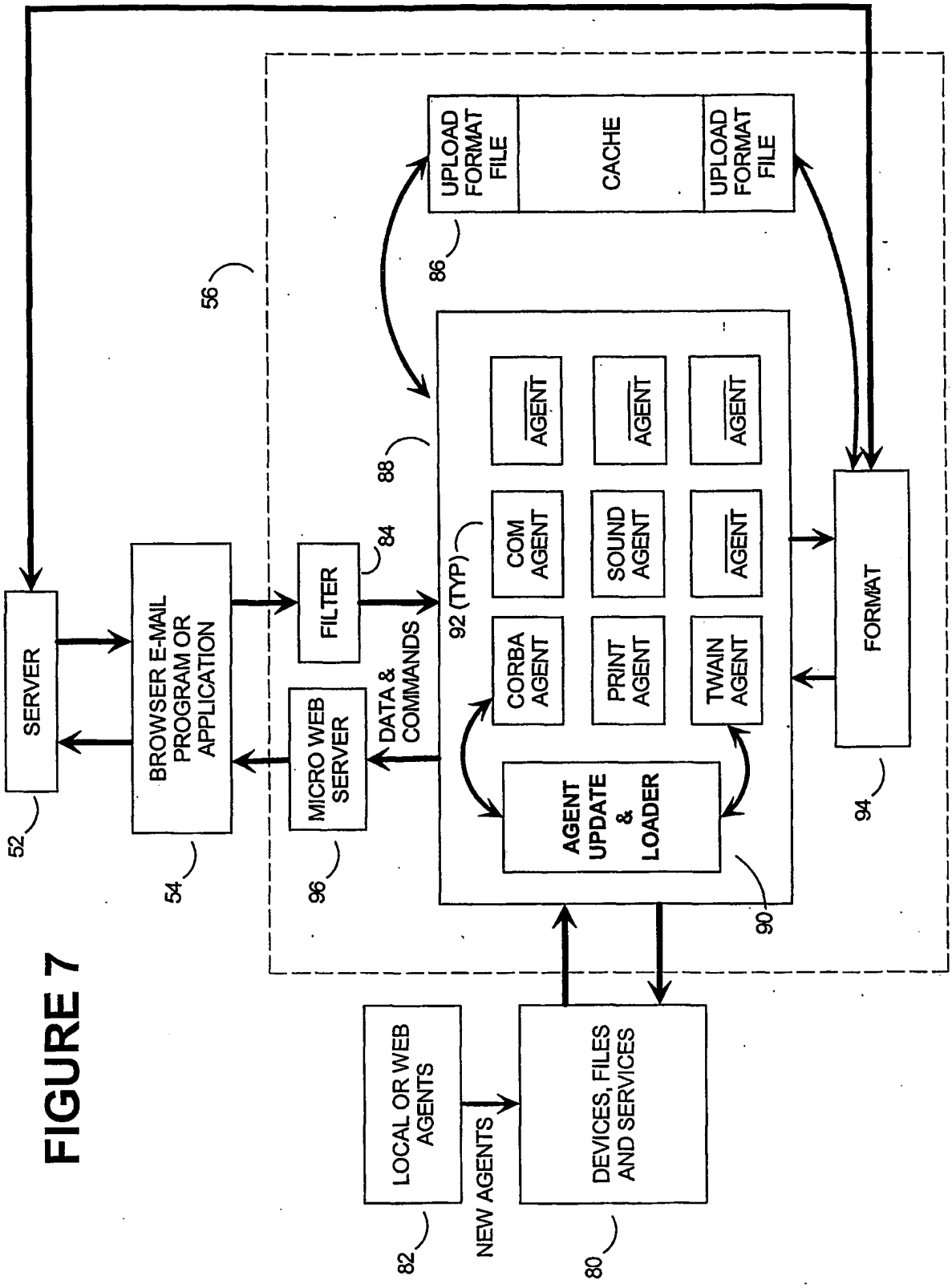

SERVER
52
BROWSER E-MAIL PROGRAM OR APPLICATION
54
FILTER
84
MICRO WEB SERVER
96
DATA & COMMANDS
56
UPLOAD FORMAT FILE
CACHE
UPLOAD FORMAT FILE
86
88
92 (TYP)
AGENT
AGENT
AGENT
COM AGENT
SOUND AGENT
AGENT
CORBA AGENT
PRINT AGENT
TWAIN AGENT
AGENT UPDATE & LOADER
90
FORMAT
94
LOCAL OR WEB AGENTS
82
NEW AGENTS
DEVICES, FILES AND SERVICES
80

FIGURE 7